International Conference of Computational Methods in Sciences and Engineering (ICCMSE 2003)
September 12-16, 2003, Kastoria, Greece, pp.484-489
World Scientific Publishing Co., River Edge, NJ, U.S.A., 2003 [ISBN: 981-238-595-9]

COMPUTING NASH EQUILIBRIA THROUGH PARTICLE SWARM OPTIMIZATION

N.G. PAVLIDIS, K.E. PARSOPOULOS AND
M.N. VRAHATIS

Department of Mathematics,
University of Patras Artificial Intelligence Research Center (UPAIRC),
University of Patras, GR-26110 Patras, Greece
E-mail: {npav, elena, kostasp, vrahatis}@math.upatras.gr

This paper considers the application of a novel optimization method, namely Particle Swarm Optimization, to compute Nash equilibria. The problem of computing equilibria is formed as one of detecting the global minimizers of a real–valued, non-negative, function. To detect more than one global minimizers of the function at a single run of the algorithm and address effectively the problem of local minima, the recently proposed Deflection technique is employed. The performance of the proposed algorithm is compared to that of algorithms implemented in the popular game theory software suite, GAMBIT. Conclusions are derived.

1. Introduction

A central solution concept in game theory is that of *Nash equilibrium*.[5] Several approaches have been proposed for the computation of Nash equilibria in finite strategic games but computing such solutions remains a challenging task. Furthermore, as pointed out in Ref. 3, computing a single Nash equilibrium is inadequate for many applications. The problem of detecting a Nash equilibrium can be formulated as a global minimization problem. This approach enables us to consider an efficient and effective optimization method, named Particle Swarm Optimization (PSO), to address this problem. Incorporating the recently proposed Deflection technique for alleviating local minima and finding more than one global minimizers of a function, several Nash equilibria can be located in a single run of the algorithm. The performance of the proposed algorithm is compared to that of algorithms implemented in the popular game theory software suite GAMBIT.[a]

[a]The GAMBIT suite is freely available from: `http://www.hss.caltech.edu/gambit/`.

The paper is organized as follows: Section 2 is devoted to the formulation of the problem. In Sections 3 and 4 the PSO and the Deflection techniques are briefly exposed. Experimental results are reported in Section 5 and conclusions are drawn in Section 6.

2. Problem Formulation

2.1. *Strategic Games and Nash Equilibria*

Definition 2.1. A **strategic game** consists of a finite set $\mathcal{N} = \{1, \ldots, n\}$ of **players**; for each player $i \in \mathcal{N}$ a **strategy set** $S_i = \{s_{i1}, \ldots, s_{im_i}\}$ is given, consisting of m_i **pure strategies**. For each $i \in \mathcal{N}$, a **payoff function** $u_i : S \to \mathbb{R}$ is also given, where $S = \times_{i \in \mathcal{N}} S_i$ is the cartesian product of all S_i's.

Let $\mathcal{P}_i$ be the set of real valued functions on S_i. The notation $p_{ij} = p_i(s_{ij})$, is used for the elements $p_i \in \mathcal{P}_i$. Let also $\mathcal{P} = \times_{i \in \mathcal{N}} \mathcal{P}_i$ and $m = \sum_{i \in \mathcal{N}} m_i$. Then $\mathcal{P}$ is isomorphic to $\mathbb{R}^m$. We denote elements in $\mathcal{P}$ by $p = (p_1, p_2, \ldots, p_n)$, where $p_i = (p_{i1}, p_{i2}, \ldots, p_{im_i}) \in \mathcal{P}_i$. If $p \in \mathcal{P}$, and $p'_i \in \mathcal{P}_i$, we use the notation (p'_i, p_{-i}) for the element $q \in \mathcal{P}$ that satisfies $q_i = p'_i$ and $q_i = p_j$ for $j \neq i$.

Now let Δ_i be the set of probability measures on S_i. We define $\Delta = \times_{i \in \mathcal{N}} \Delta_i$, so $\Delta \subseteq \mathbb{R}^m$. Thus, the elements $p_i \in \Delta_i$ are real valued functions on S_i, $p_i : S_i \to \mathbb{R}$ and it holds that $\sum_{s_{ij} \in S_i} p_i(s_{ij}) = 1$, and $p_i(s_{ij}) \geqslant 0, \forall s_{ij} \in S_i$.

We use the abusive notation s_{ij} to denote the strategy $p_i \in \Delta_i$ with $p_{ij} = 1$. Hence, the notation (s_{ij}, p_{-i}) represents the strategy where player i adopts the pure strategy s_{ij}, and all the other players adopt their components of p.

The payoff function u is extended to have domain $\mathbb{R}^m$ by the rule

$$u_i(p) = \sum_{s \in S} p(s) u_i(s), \tag{1}$$

where we define

$$p(s) = \prod_{i \in \mathcal{N}} p_i(s_i). \tag{2}$$

Definition 2.2. A strategy $p^* = (p_1^*, p_2^*, \ldots, p_n^*) \in \mathcal{P}$ is a **Nash equilibrium** if $p^* \in \Delta$ and for all $i \in \mathcal{N}$ and all $p_i \in \Delta_i$, $u_i(p_i, p_{-i}^*) \leqslant u_i(p^*)$.

2.2. *Nash Equilibrium as a Global Minimizer*

To formulate the problem of finding a Nash equilibrium to that of detecting the global minimum of a real valued function, three functions, x, z and

$g : \mathcal{P} \to \mathbb{R}^m$, are required. For any $p \in \mathcal{P}$, $i \in N$ and $s_{ij} \in S_i$, define:

$$x_{ij}(p) = u_i(s_{ij}, p_{-i}), \tag{3}$$

$$z_{ij}(p) = x_{ij}(p) - u_i(p), \tag{4}$$

$$g_{ij}(p) = \max[z_{ij}(p), 0]. \tag{5}$$

The real valued function $v : \mathcal{P} \to \mathbb{R}$, is defined as:

$$v(p) = \sum_{i \in \mathcal{N}} \sum_{1 \leqslant j \leqslant m_i} [g_{ij}(p)]^2. \tag{6}$$

Function v is nonnegative and continuously differentiable. Furthermore, p^* is a Nash equilibrium if and only if, it is a global minimizer of v, i.e. $v(p^*) = 0$, and $p^* \in \Delta$.

3. Particle Swarm Optimization

PSO belongs to the broad class of stochastic optimization algorithms. PSO is a population–based algorithm that exploits a population of individuals, to probe promising regions of the search space. In this context, the population is called *swarm* and the individuals are called *particles*. Each particle is assigned to a neighborhood and moves with an adaptable velocity within the search space, retaining in its memory the best position it ever encountered. Moreover, the best position ever attained by all individuals of a neighborhood is communicated to the particles that comprise the neighborhood.[7]

Assume a D–dimensional search space, $S \subset \mathbb{R}^D$, and a swarm consisting of N particles. The i–th particle is in effect a D–dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})^\top$. The velocity of this particle is also a D–dimensional vector, $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})^\top$. The best previous position encountered by the i–th particle is a point in S, denoted as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})^\top$. Assume g to be the index of the particle that attained the best previous position among all the individuals of the swarm, and t to be the iteration counter. Then, according to the latest version of PSO, which incorporates a parameter called *constriction factor*, the swarm is manipulated using the following equations:[1]

$$V_i(t+1) = \chi\Big(V_i(t) + c_1 r_1\big(P_i(t) - X_i(t)\big) + c_2 r_2\big(P_g(t) - X_i(t)\big)\Big), \tag{7}$$

$$X_i(t+1) = X_i(t) + V_i(t+1), \tag{8}$$

where $i = 1, 2, \ldots, N$; χ is the constriction factor; c_1 and c_2 are positive parameters called *cognitive* and *social* parameter respectively; r_1, r_2 are random numbers uniformly distributed in the interval $[0, 1]$; and t, stands for the counter of iterations. The formulae used for the computation of the constriction factor's value are reported in Ref. 1.

4. Detecting Several Minimizers Through Deflection

To detect several global minimizers in a single run, the *Deflection* technique is applied. Let $f : S \to \mathbb{R}$, $S \subset \mathbb{R}^n$, be the original objective function under consideration. Let also x_i^*, $i = 1, \ldots, m$, be m minimizers of f. Then, the *Deflection* technique is defined as:

$$F(x) = T_1(x; x_1, \lambda_1)^{-1} \cdots T_m(x; x_m, \lambda_m)^{-1} f(x), \tag{9}$$

where λ_i, $i = 1, \ldots, m$, are relaxation parameters. The functions,

$$T_i(x; x_i, \lambda_i) = \tanh(\lambda_i \|x - x_i^*\|), \quad i = 1, \ldots, m, \tag{10}$$

satisfy the property, that any sequence of points $\{x_k\}_{k=0}^{\infty}$ converging to any one of the minimizers x_i^*, does not produce a minimum of F at $x = x_i^*$, while all other minima of f remain unaffected, as shown in Ref. 2. Note that if the global minimum of the function is zero, a function $\hat{f} = f + c$, where $c > 0$ is a constant, should replace f in Eq. (9).

5. Experimental Results

The proposed algorithm has been applied on noncooperative strategic games characterized by more than one Nash equilibrium. A comparison of the algorithm's performance against the Lyapunov function algorithm implemented in GAMBIT produced promising results. The particular algorithm was selected since it is the suggested algorithm for detecting all the equilibria of an n–person game.[4] Indicative test problems are defined below:

Test problem 1 [BACH OR STRAVINSKY[6]]: Two–player game, with two pure strategies available to each player and 3 equilibria. The payoff matrix for this game is illustrated in Table 1.

Test problem 2 [HAWK–DOVE[6]]: Two–player game, with two pure strategies available to each player and 3 equilibria. The payoff matrix for this game is also illustrated in Table 1.

Test problem 3 [STAG HUNT GAME[4]]: Two–player game, with three pure strategies available to each player and 6 equilibria. The payoff matrix is illustrated in Table 2.

Test problem 4: Random strategic game with two players and three pure strategies available to each player, with payoffs randomly distributed in the interval $[0, 1]$ and 3 equilibria. The payoff matrix is illustrated in Table 3.

Test problem 5: Normal form three player game with two pure strategies available to each player and 9 equilibria.[3] The payoffs of this game are given in Table 4.

Table 1. Payoff matrices for Test Problems 1 (left) and 2 (right).

	Bach	Stravinsky
Bach	2, 1	0, 0
Stravinsky	0, 0	1, 2

	Dove	Hawk
Dove	3, 3	1, 4
Hawk	4, 1	0, 0

Table 2. Payoff matrix for Test Problem 3.

	s_{21}	s_{22}	s_{23}
s_{11}	0.5, 0.5	0.5, 0	0.5, −0.5
s_{12}	0, 0.5	1, 1	1, 0.5
s_{13}	−0.5, 0.5	0.5, 1	1.5, 1.5

Table 3. Payoff matrix for Test Problem 4.

	s_{21}	s_{22}	s_{23}
s_{11}	0.2190, 0.0535	0.6793, 0.0077	0.5194, 0.4175
s_{12}	0.0470, 0.5297	0.9347, 0.3834	0.8310, 0.6868
s_{13}	0.6789, 0.6711	0.3835, 0.0668	0.0346, 0.5890

Table 4. Payoff matrix for Test Problem 5.

s_{31}	s_{21}	s_{22}
s_{11}	9, 8, 12	0, 0, 0
s_{12}	0, 0, 0	9, 8, 2

s_{32}	s_{21}	s_{22}
s_{11}	0, 0, 0	3, 4, 6
s_{12}	3, 4, 4	0, 0, 0

The configuration of the PSO parameters has been fixed for all test problems, with the exception of swarm size, which was problem dependent.

Table 5. Experimental Results.

Test Problem	Nash Equilibria	PSO		Lyapunov Function	
		SR	FE	SR	FE
1	3	100%	530	50%	2148
2	3	100%	884	30%	1188
3	6	80%	4050	0%	14835
4	3	100%	7500	10%	177536
5	9	82%	6700	0%	75583

Thus, for the constriction factor χ and the cognitive and social parameter, c_1 and c_2, respectively, the default values $\chi = 0.729$, $c_1 = c_2 = 2.05$ have been used.[1] For the Deflection technique, the setup $\gamma_1 = \frac{10^4}{2}$, $\gamma_2 = \frac{1}{2}$, $\mu = 10^{-10}$, and $\lambda = 1$, respectively, was selected. The desired accuracy for detecting a Nash equilibrium has been equal to 10^{-6} in all cases. The obtained results are exhibited in Table 5. In particular, the success rate (SR) of the PSO and the Lyapunov function method on locating *all* Nash equilibria averaged over 10 runs, as well as the mean function evaluations required for the computation of one equilibrium (FE), are reported.

6. Conclusions

In this contribution the problem of detecting the Nash equilibria of finite strategic games was addressed through the Particle Swarm Optimization method, equipped with the Deflection technique. Deflection enables the algorithm to overcome local minima and detect more than one global minimizers at a single run. Numerical experiments performed on a number of finite strategic games suggest that the particular approach addresses the problem effectively.

References

1. M. Clerc, J. Kennedy, *IEEE Trans. Evol. Comput.*, 6(1):58–73, 2001.
2. G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, *Nonlinear Analysis, Theory, Methods & Applications*, 30(7):4545–4550, 1997.
3. R.D. McKelvey, A. McLennan, In H.M. Amman, D.A. Kendrick, and J. Rust (Eds.), *Handbook of Computational Economics*, pp. 87–142. North–Holland, 1996.
4. R.D. McKelvey, A. McLennan, T. Turocy, *Manual of the Gambit Command Language*, Ver. 0.96.3, California Institute of Technology, 2000.
5. J.F. Nash, *Annals of Mathematics*, 54:289–295, 1951.
6. M.J. Osborne, A. Rubinstein, *A Course in Game Theory*, 1994, MIT Press.
7. K.E. Parsopoulos, M.N. Vrahatis, *Natural Computing*, 1(2–3):235–306, 2002.